

**DERİN ÖĞRENME VE EVRİŞİMLİ SİNİR AĞLARI**

**2. ÖDEV RAPORU**

**G191210386**

**YASİN ALTUNBAŞAK**

Giriş:

Bu projede Derin öğrenmenin özelleşmiş yapılarından evrişimsel sinir ağları (CNN) ve uygulama olarak Google Colaboratory (Colab) kullanıldı. CNN, katmanlı yapısıyla gizli nitelikleri ortaya çıkararak işlem yapmaktadır. Gizli katman sayısının artışı bir noktaya kadar olumlu etki yaparken sonrasında parametre artışından dolayı bulma hızını düşürdüğü için performansa olumsuz yönde etkilemektedir. 6 katman (4 konvolüsyon katmanı + 2 tam bağlantı katmanı (Dense Layer) ) her bir katmandaki hücre sayısı 16-256 arasında olacak şekilde kullanıldı.

Blokların Açıklanması:

[1] kod bloğumuzda gerekli kütüphaneleri çağırdık.(tensorflow sınıfını çağırırken sıkıntı yaşadım)

[2] Cifar100 veri setini yükledim.

Downloading data from <https://www.cs.toronto.edu/~kriz/cifar-100-python.tar.gz>

169009152/169001437 [==============================] - 2s 0us/step

169017344/169001437 [==============================] - 2s 0us/step

(50000, 32, 32, 3) (50000, 1)

(10000, 32, 32, 3) (10000, 1)

[3] CIFAR-100 veri setinden öğrenci numarama göre belirlenen 7 farklı sınıfı (17: "castle",22: "clock",4: "lamp",51: "mushroom",59: "pine\_tree",81: "streetcar",94: "wardrobe") kullandım. Daha sonra kodlanan save\_resimler() fonksiyon ile training ve test verileri olarak kaydedildi. Projede os.mkdir (veriseti\_path) ve os.path kullanıldı. Python'da klasörlere ilişkin fonksiyonları os modülünün path nesnesinde bulabilir.

[4] “epoch” eğitimi yaptırdım ve her iterasyon için 50 adet fotoğraf aldım.

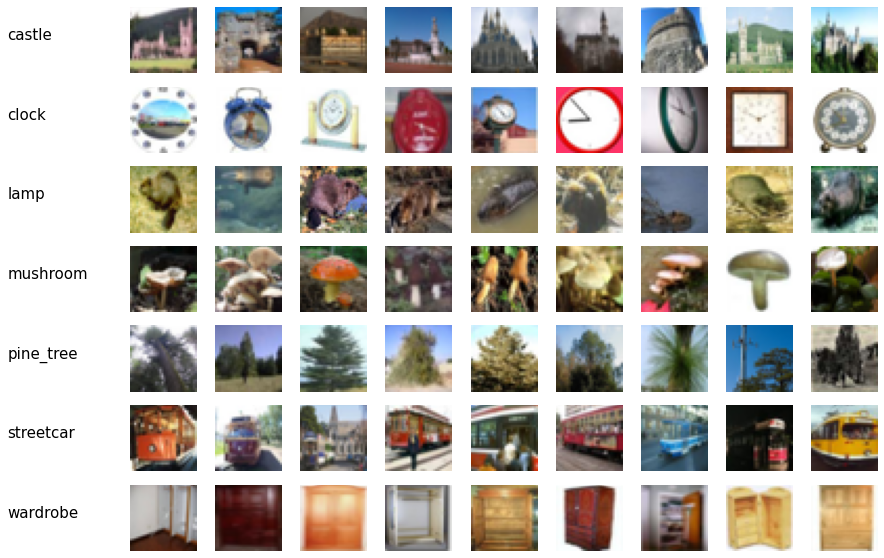
[5] Veri setimde bulunan 7 sınıfı len fonksiyonu ile aldım.

[6]-[7] Daha fazla veri örneğine ihtiyacımız olduğu için Data Augmentation metodunu ekledim. Bu metod model başarımı(accuarcy) arttırmayı sağlayacak ve ezberlemeyi(overfitting) de önleyecektir.

Found 3500 images belonging to 7 classes.

Found 700 images belonging to 7 classes.

[8] get\_image\_via\_class() oluşturduğum bu sınıf ile elimdeki verileri eğittim. Ve elimize aşağıdaki gibi bir çıktı verdi.



[9] Bu kısımda Relu activation fonksiyonunu kullanan bir katman ekledim. 32 adet 3\*3 boyutunda, Relu activation fonksiyonunu kullanan bir katman daha ekledim. 2\*2 çerçeveden oluşan MaxPooling katmanını ekledim. Modelin aşırı ezberlemesine engel olmak için verilerin %25’ini Dropout katmanı ile kapattım.

64 adet 3\*3 boyutunda, Relu activation fonksiyonunu kullanan bir katman ekledim. 2\*2 çerçeveden oluşan MaxPooling katmanını ekledim.Modelin aşırı ezberlemesine engel olmak için verilerin %50’sini Dropout katmanı ile kapattım.

128 adet 3\*3 boyutunda, Relu activation fonksiyonunu kullanan bir katman ekledim. 2\*2 çerçeveden oluşan MaxPooling katmanını ekledim.

Flatten katmanı yardımı ile 2 boyutlu görüntüleri 1 boyutlu vektörlere çevirdim. Modelin aşırı ezberlemesine engel olmak için verilerin %50’sini Dropout katmanı ile kapattım.

256 adet nörondan oluşan Dense katmanını ekledim.

Son olarak da çıkış katmanı ekledim. Bu katmanda Elimdeki sınıfların sayısı kadar nöron vardır ve softmax fonksiyonu kullanmaktadır.

[10] 9. Blokta oluşturduğum modeli eğitim sınıfı içerisine yerleştirdim.

[11] Bu blokta “historyList”in grafiğini oluşturacak loss\_acc\_graph sınıfı yazdım.

[13] bu blokta tenserflowun kütüphanesiyle yaşadığım sorundan kaynaklı olarak ciddi bir problem yaşadım ve Grafikleri ekrana verecek diğer fonksiyonların oluşmasına engel oldu. Maalesef bu kısımdan sonrası için çözüm bulamadım.

Zorlandığım kısımlar:

Python dilinin kendi gramerine alışık olmadığım için adapte olmakta biraz zorlandım. Kütüphanelerin import ederken halen daha anlayamadığım sorunlar projeyi tamamen bitirmeme engel oldu.